What is "**Embedded - Microcontrollers**"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "**Embedded - Microcontrollers**"

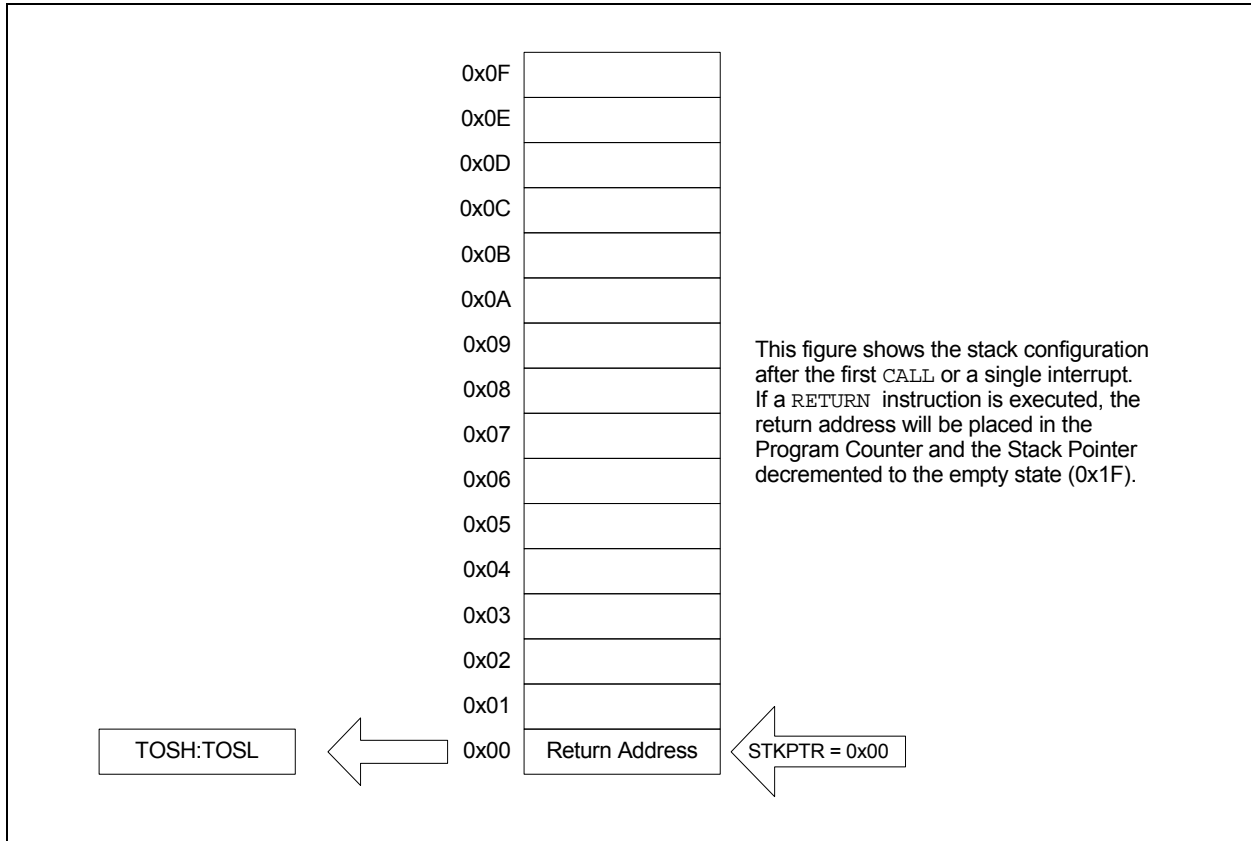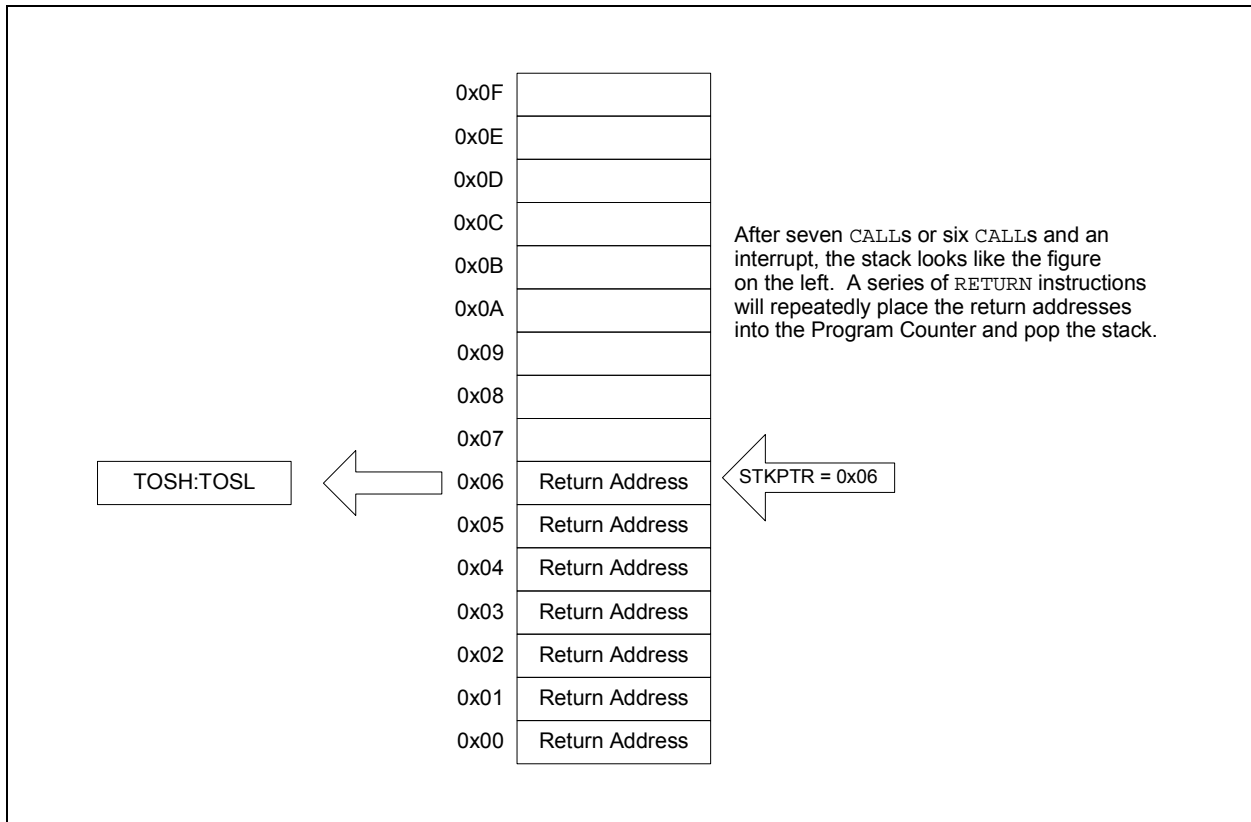| Details	
Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	20MHz
Connectivity	I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	25
Program Memory Size	3.5KB (2K x 14)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	128 x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 17x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SOIC (0.295", 7.50mm Width)
Supplier Device Package	28-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16lf1512-i-so

FIGURE 3-6: **ACCESSING THE STACK EXAMPLE 2**

0x0F	
0x0E	
0x0D	
0x0C	
0x0B	
0x0A	
0x09	
0x08	
0x07	
0x06	
0x05	
0x04	
0x03	
0x02	
0x01	
0x00	Return Address

TOSH:TOSL ← 0x00 Return Address ← STKPTR = 0x00

This figure shows the stack configuration after the first CALL or a single interrupt. If a RETURN instruction is executed, the return address will be placed in the Program Counter and the Stack Pointer decremented to the empty state (0x1F).

FIGURE 3-7: **ACCESSING THE STACK EXAMPLE 3**

0x0F	
0x0E	
0x0D	
0x0C	
0x0B	
0x0A	
0x09	
0x08	
0x07	
0x06	Return Address
0x05	Return Address
0x04	Return Address
0x03	Return Address
0x02	Return Address
0x01	Return Address
0x00	Return Address

TOSH:TOSL ← 0x06 Return Address ← STKPTR = 0x06

After seven CALLs or six CALLs and an interrupt, the stack looks like the figure on the left. A series of RETURN instructions will repeatedly place the return addresses into the Program Counter and pop the stack.

REGISTER 4-2: **CONFIGURATION WORD 2**

R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	U-1
LVP	$\overline{\text{DEBUG}}$	$\overline{\text{LPBOR}}$	BORV	STVREN	—
bit 13					bit 8

U-1	U-1	U-1	R/P-1	U-1	U-1	R/P-1	R/P-1
—	—	—	$\overline{\text{VCAPEN}}$[(1)]	—	—	WRT<1:0>	
bit 7							bit 0

Legend:		
R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as '1'
'0' = Bit is cleared	'1' = Bit is set	-n = Value when blank or after Bulk Erase

bit 13 **LVP:** Low-Voltage Programming Enable bit
1 = Low-voltage programming enabled
0 = High-voltage on $\overline{\text{MCLR}}$ must be used for programming

bit 12 **DEBUG:** In-Circuit Debugger Mode bit
1 = In-Circuit Debugger disabled, ICSPCLK and ICSPDAT are general purpose I/O pins
0 = In-Circuit Debugger enabled, ICSPCLK and ICSPDAT are dedicated to the debugger

bit 11 **LPBOR:** Low-Power BOR bit
1 = Low-Power BOR is disabled
0 = Low-Power BOR is enabled

bit 10 **BORV:** Brown-out Reset Voltage Selection bit[(2)]
1 = Brown-out Reset voltage (V$_{BOR}$), low trip point selected
0 = Brown-out Reset voltage (V$_{BOR}$), high trip point selected

bit 9 **STVREN:** Stack Overflow/Underflow Reset Enable bit
1 = Stack Overflow or Underflow will cause a Reset
0 = Stack Overflow or Underflow will not cause a Reset

bit 8-5 **Unimplemented:** Read as '1'

bit 4 **VCAPEN:** Voltage Regulator Capacitor Enable bits[(1)]
If PIC16LF1512/3 (regulator disabled):
   These bits are ignored. All V$_{CAP}$ pin functions are disabled.
If PIC16F1512/3 (regulator enabled):
   0 = V$_{CAP}$ functionality is enabled on RA5
   1 = All V$_{CAP}$ pin functions are disabled

bit 3-2 **Unimplemented:** Read as '1'

bit 1-0 **WRT<1:0>:** Flash Memory Self-Write Protection bits
2 kW Flash memory (PIC16(L)F1512 only):
   11 = Write protection off
   10 = 000h to 1FFh write-protected, 200h to 7FFh may be modified by PMCON control
   01 = 000h to 3FFh write-protected, 400h to 7FFh may be modified by PMCON control
   00 = 000h to 7FFh write-protected, no addresses may be modified by PMCON control
4 kW Flash memory (PIC16(L)F1513 only):
   11 = Write protection off
   10 = 000h to 1FFh write-protected, 200h to FFFh may be modified by PMCON control
   01 = 000h to 7FFh write-protected, 800h to FFFh may be modified by PMCON control
   00 = 000h to FFFh write-protected, no addresses may be modified by PMCON control

Note 1: PIC16F1512/3 only.
2: See V$_{BOR}$ parameter for specific trip point voltages.

FIGURE 11-5: **BLOCK WRITES TO FLASH PROGRAM MEMORY WITH 32 WRITE LATCHES**
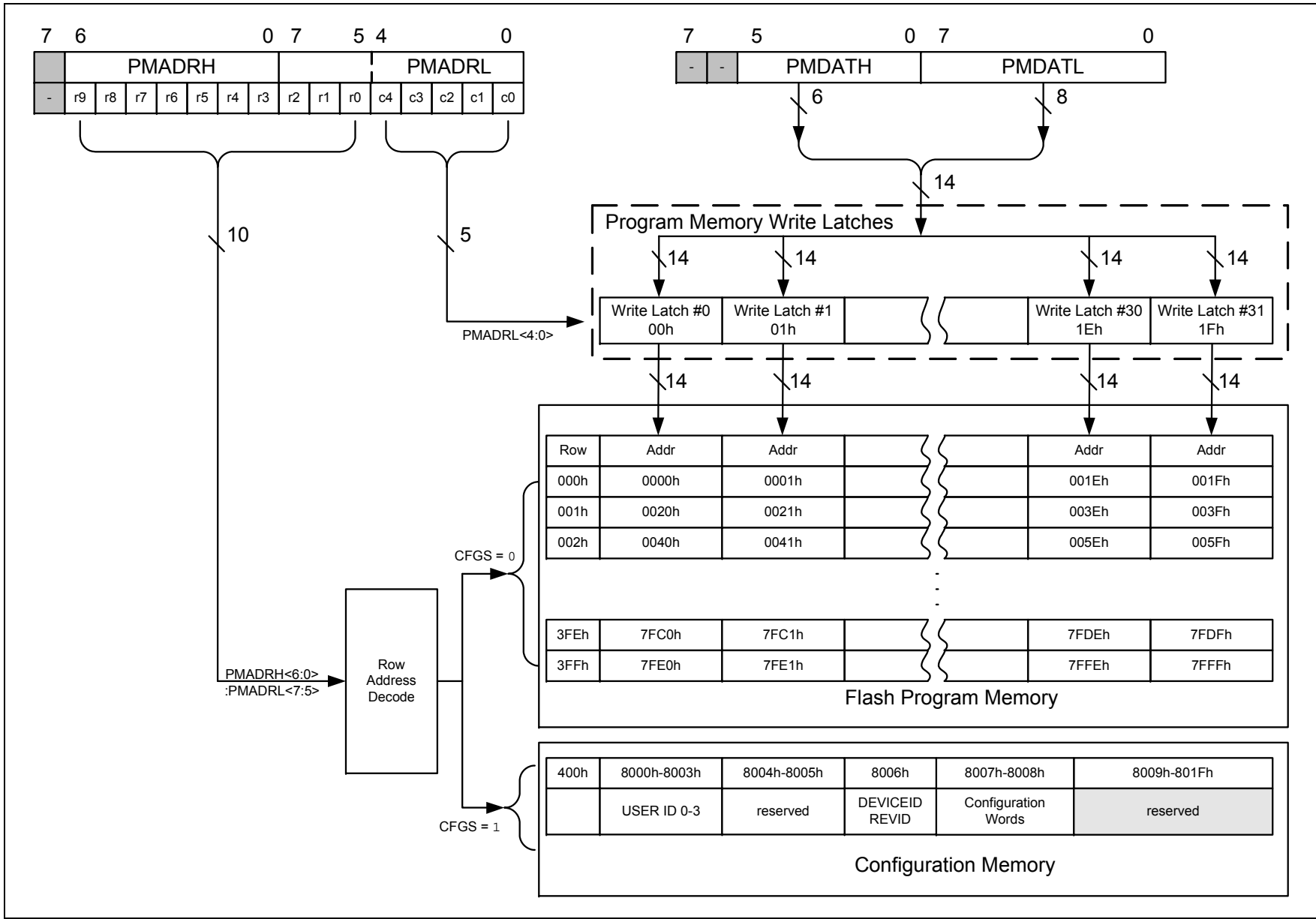
PIC16(L)F1512/3

12.0 I/O PORTS

Each port has three standard registers for its operation. These registers are:

- TRISx registers (data direction)
- PORTx registers (reads the levels on the pins of the device)
- LATx registers (output latch)

Some ports may have one or more of the following additional registers. These registers are:

- ANSELx (analog select)
- WPUx (weak pull-up)

In general, when a peripheral is enabled on a port pin, that pin cannot be used as a general purpose output. However, the pin can still be read.

TABLE 12-1: PORT AVAILABILITY PER DEVICE

Device	PORTA	PORTB	PORTC	PORTE
PIC16(L)F1512	●	●	●	●
PIC16(L)F1513	●	●	●	●

The Data Latch (LATx registers) is useful for read-modify-write operations on the value that the I/O pins are driving.

A write operation to the LATx register has the same effect as a write to the corresponding PORTx register. A read of the LATx register reads of the values held in the I/O PORT latches, while a read of the PORTx register reads the actual I/O pin value.

Ports that support analog inputs have an associated ANSELx register. When an ANSEL bit is set, the digital input buffer associated with that bit is disabled. Disabling the input buffer prevents analog signal levels on the pin between a logic high and low from causing excessive current in the logic input circuitry. A simplified model of a generic I/O port, without the interfaces to other peripherals, is shown in Figure 12-1.

FIGURE 12-1: GENERIC I/O PORT OPERATION
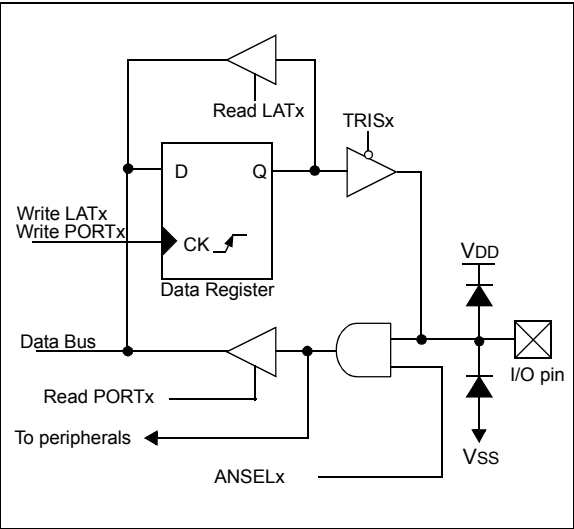
EXAMPLE 12-1: INITIALIZING PORTA

```
; This code example illustrates
; initializing the PORTA register. The
; other ports are initialized in the same
; manner.

BANKSEL  PORTA         ;
CLRF     PORTA         ;Init PORTA
BANKSEL  LATA          ;Data Latch
CLRF     LATA          ;
BANKSEL  ANSELA        ;
CLRF     ANSELA        ;digital I/O
BANKSEL  TRISA         ;
MOVLW    B'00111000'   ;Set RA<5:3> as inputs
MOVWF    TRISA         ;and set RA<2:0> as
                       ;outputs
```

12.2 PORTA Registers

PORTA is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISA (Register 12-3). Setting a TRISA bit (= 1) will make the corresponding PORTA pin an input (i.e., disable the output driver). Clearing a TRISA bit (= 0) will make the corresponding PORTA pin an output (i.e., enables output driver and puts the contents of the output latch on the selected pin). Example 12-1 shows how to initialize PORTA.

Reading the PORTA register (Register 12-2) reads the status of the pins, whereas writing to it will write to the PORT latch. All write operations are read-modify-write operations. Therefore, a write to a port implies that the port pins are read, this value is modified and then written to the PORT data latch (LATA).

The TRISA register (Register 12-3) controls the PORTA pin output drivers, even when they are being used as analog inputs. The user should ensure the bits in the TRISA register are maintained set when using them as analog inputs. I/O pins configured as analog input always read '0'.

12.2.1 ANSELA REGISTER

The ANSELA register (Register 12-5) is used to configure the Input mode of an I/O pin to analog. Setting the appropriate ANSELA bit high will cause all digital reads on the pin to be read as '0' and allow analog functions on the pin to operate correctly.

The state of the ANSELA bits has no effect on digital output functions. A pin with TRIS clear and ANSEL set will still operate as a digital output, but the Input mode will be analog. This can cause unexpected behavior when executing read-modify-write instructions on the affected port.

> **Note:** The ANSELA bits default to the Analog mode after Reset. To use any pins as digital general purpose or peripheral inputs, the corresponding ANSEL bits must be initialized to '0' by user software.

12.2.2 PORTA FUNCTIONS AND OUTPUT PRIORITIES

Each PORTA pin is multiplexed with other functions. The pins, their combined functions and their output priorities are shown in Table 12-2.

When multiple outputs are enabled, the actual pin control goes to the peripheral with the highest priority.

Analog input functions, such as ADC, are not shown in the priority lists. These inputs are active when the I/O pin is set for Analog mode using the ANSELx registers. Digital output functions may control the pin when it is in Analog mode with the priority shown in Table 12-2.

TABLE 12-2: PORTA OUTPUT PRIORITY

Pin Name	Function Priority[1]
RA0	RA0
RA1	RA1
RA2	RA2
RA3	RA3
RA4	RA4
RA5	V_{CAP} (PIC16F1512/3 only) RA5
RA6	CLKOUT OSC2 RA6
RA7	RA7

Note 1: Priority listed from highest to lowest.

REGISTER 12-17: WPUE: WEAK PULL-UP PORTE REGISTER[1,2]

U-0	U-0	U-0	U-0	R/W-1/1	U-0	U-0	U-0
—	—	—	—	WPUE3	—	—	—
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	

bit 7-4   **Unimplemented:** Read as '0'

bit 3   **WPUE:** Weak Pull-up Register bit
             1 = Pull-up enabled
             0 = Pull-up disabled

bit 2-0   **Unimplemented:** Read as '0'

Note 1: Global $\overline{\text{WPUEN}}$ bit of the OPTION_REG register must be cleared for individual pull-ups to be enabled.

2: The weak pull-up device is automatically disabled if the pin is in configured as an output.

TABLE 12-9:   SUMMARY OF REGISTERS ASSOCIATED WITH PORTE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
A(A)DCON0	—	CHS<4:0>					GO/DONE	ADON	130, 147
CCPxCON	—	—	DCxB<1:0>		CCPxM<3:0>				236
PORTE	—	—	—	—	RE3	—	—	—	113
TRISE	—	—	—	—	—(1)	—	—	—	113
WPUE	—	—	—	—	WPUE3	—	—	—	114

Legend: x = unknown, u = unchanged, – = unimplemented locations read as '0'. Shaded cells are not used by PORTE.

Note 1:   Unimplemented, read as '1'.

TABLE 12-10:   SUMMARY OF CONFIGURATION WORD WITH PORTE

Name	Bits	Bit -/7	Bit -/6	Bit 13/5	Bit 12/4	Bit 11/3	Bit 10/2	Bit 9/1	Bit 8/0	Register on Page
CONFIG1	13:8	—		FCMEN	IESO	CLKOUTEN	BOREN<1:0>		—	37
	7:0	$\overline{\text{CP}}$	MCLRE	$\overline{\text{PWRTE}}$	WDTE<1:0>		FOSC<2:0>			

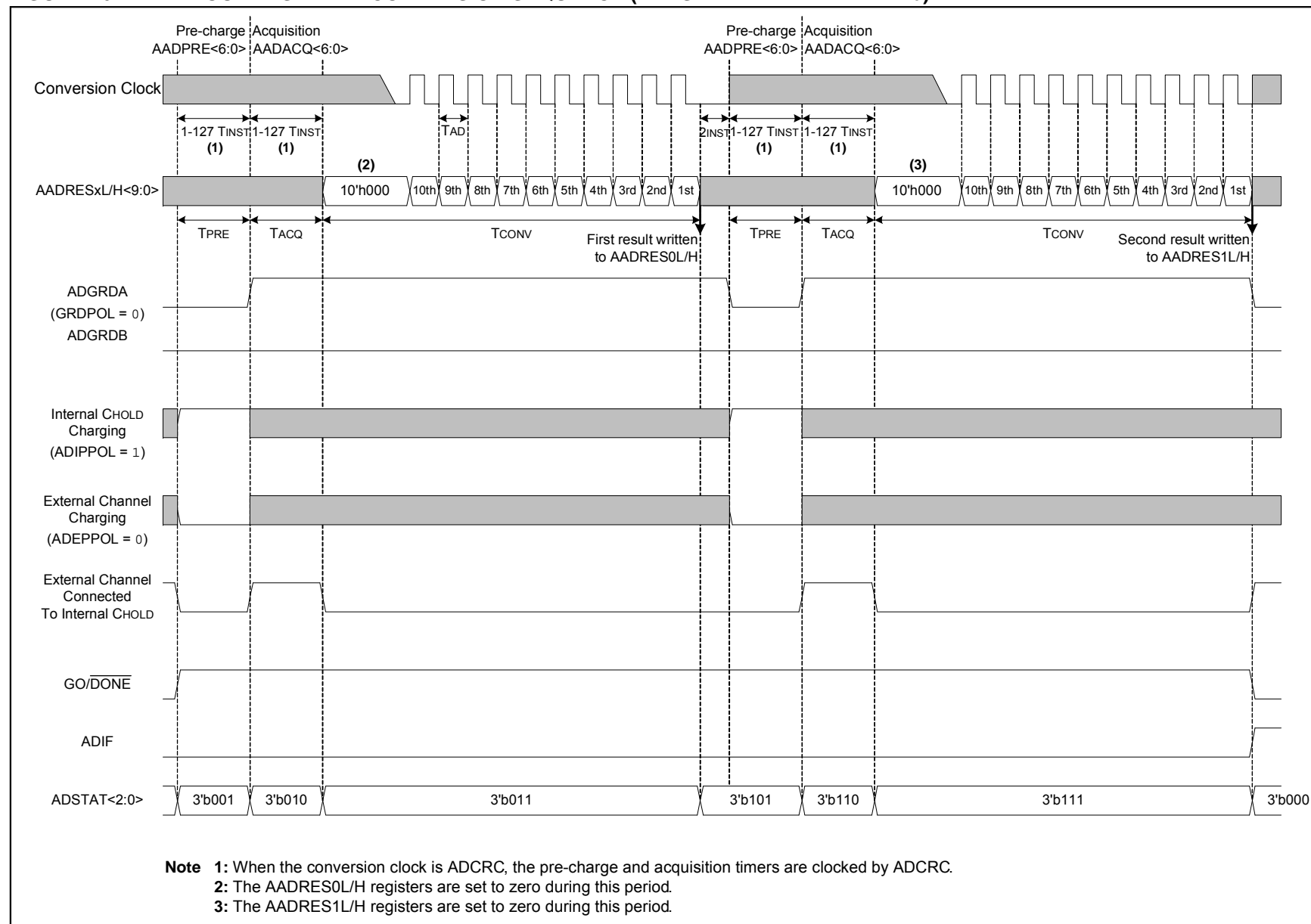**Legend:**   — = unimplemented location, read as '0'. Shaded cells are not used by PORTE.

16.0 ANALOG-TO-DIGITAL CONVERTER (ADC) MODULE

Note:	This section of the ADC chapter discusses legacy operation. If new Capacitive Voltage Divider (CVD) features are needed, refer to **Section 16.5 "Hardware Capacitive Voltage Divider (CVD) Module"** for more information.

The Analog-to-Digital Converter (ADC) allows conversion of an analog input signal to a 10-bit binary representation of that signal. This device uses analog inputs, which are multiplexed into a single sample and hold circuit. The output of the sample and hold is connected to the input of the converter. The converter generates a 10-bit binary result via successive approximation and stores the conversion result into the ADC result registers (ADRESH:ADRESL register pair). Figure 16-1 shows the block diagram of the ADC.

The ADC voltage reference is software selectable to be either internally generated or externally supplied.

The ADC can generate an interrupt upon completion of a conversion. This interrupt can be used to wake up the device from Sleep.

FIGURE 16-11: DOUBLE SAMPLE CONVERSION SEQUENCE (ADDSEN = 1 AND ADIPEN = 0)

Note 1: When the conversion clock is ADCRC, the pre-charge and acquisition timers are clocked by ADCRC.

2: The AADRES0L/H registers are set to zero during this period.

3: The AADRES1L/H registers are set to zero during this period.

PIC16(L)F1512/3

TABLE 16-5: SUMMARY OF REGISTERS ASSOCIATED WITH HARDWARE CVD

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
AADCAP	—	—	—	—	—	ADDCAP<2:0>			152
AADCON0	—	CHS<4:0>					GO/$\overline{\text{DONE}}$	ADON	147
AADCON1	ADFM	ADCS<2:0>			—	—	ADPREF<1:0>		148
AADCON2	—	TRIGSEL<2:0>			—	—	—	—	149
AADCON3	ADEPPOL	ADIPPOL	ADOLEN	ADOEN	ADOOEN	—	ADIPEN	ADDSEN	150
AADGRD	GRDBOE	GRDAOE	GRDPOL	—	—	—	—	—	152
AADPRE	—	ADPRE<6:0>							151
AADRES0H	A/D Result 0 Register High								154, 155
AADRES0L	A/D Result 0 Register Low								154, 155
AADSTAT	—	—	—	—	—	ADCONV	ADSTG<1:0>		151
AADACQ	—	ADACQ<6:0>							152
ANSELA	—	—	ANSA5	—	ANSA3	ANSA2	ANSA1	ANSA0	104
ANSELB	—	—	ANSB5	ANSB4	ANSB3	ANSB2	ANSB1	ANSB0	108
ANSELC	ANSC7	ANSC6	ANSC5	ANSC4	ANSC3	ANSC2	—	—	111
CCP1CON	—	—	DC1B<1:0>		CCP1M<3:0>				236
CCP2CON	—	—	DC2B<1:0>		CCP2M<3:0>				236
FVRCON	FVREN	FVRRDY	TSEN	TSRNG	—	—	ADFVR<1:0>		120
INTCON	GIE	PEIE	TMR0IE	INTE	IOCIE	TMR0IF	INTF	IOCIF	69
PIE1	TMR1GIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	70
PIR1	TMR1GIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	72
TRISA	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	103
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	107
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	110

Legend: — = unimplemented read as '0'. Shaded cells are not used for ADC module.

PIC16(L)F1512/3

FIGURE 20-16: I²C SLAVE, 7-BIT ADDRESS, RECEPTION (SEN = 0, AHEN = 1, DHEN = 1)
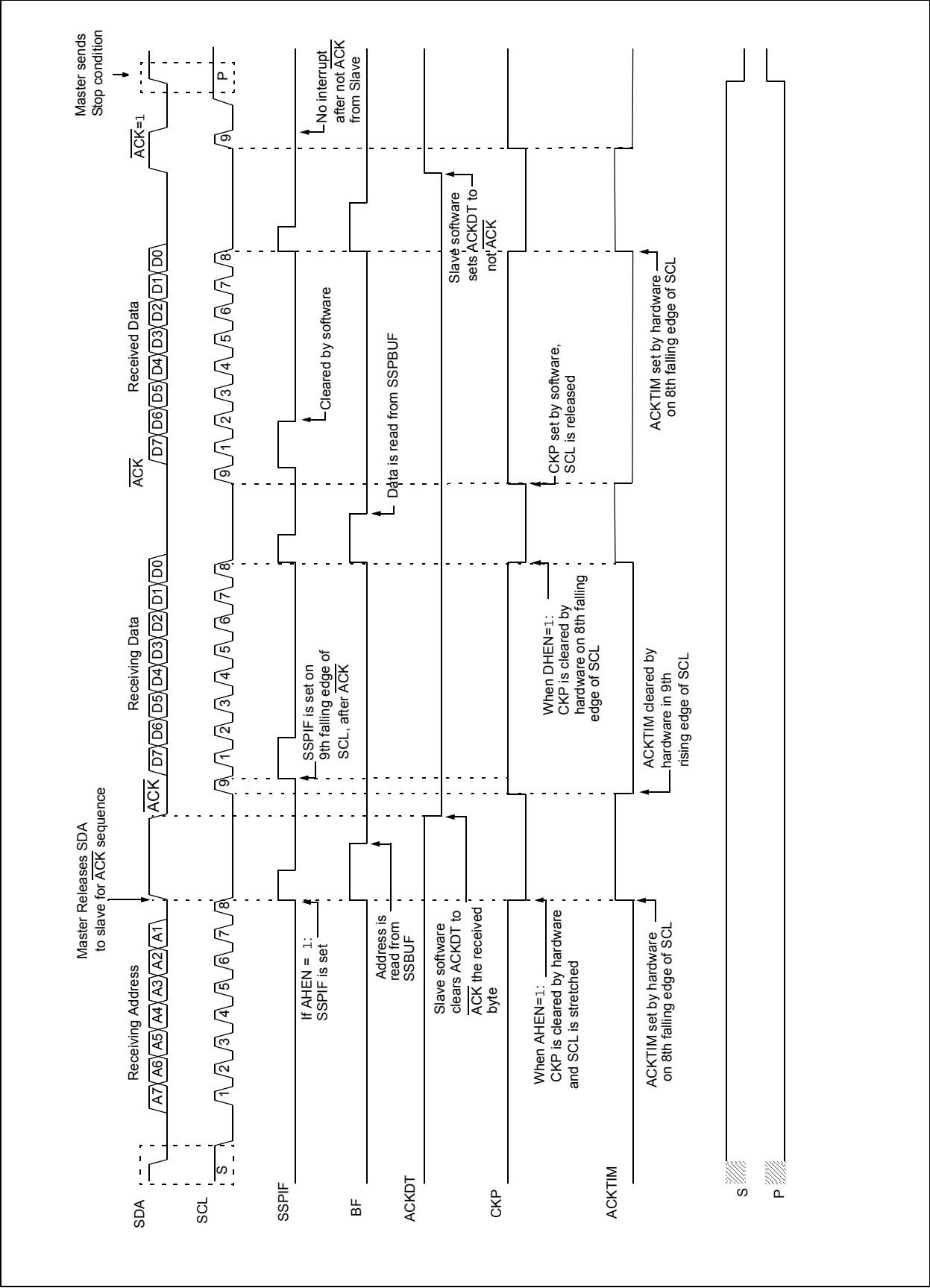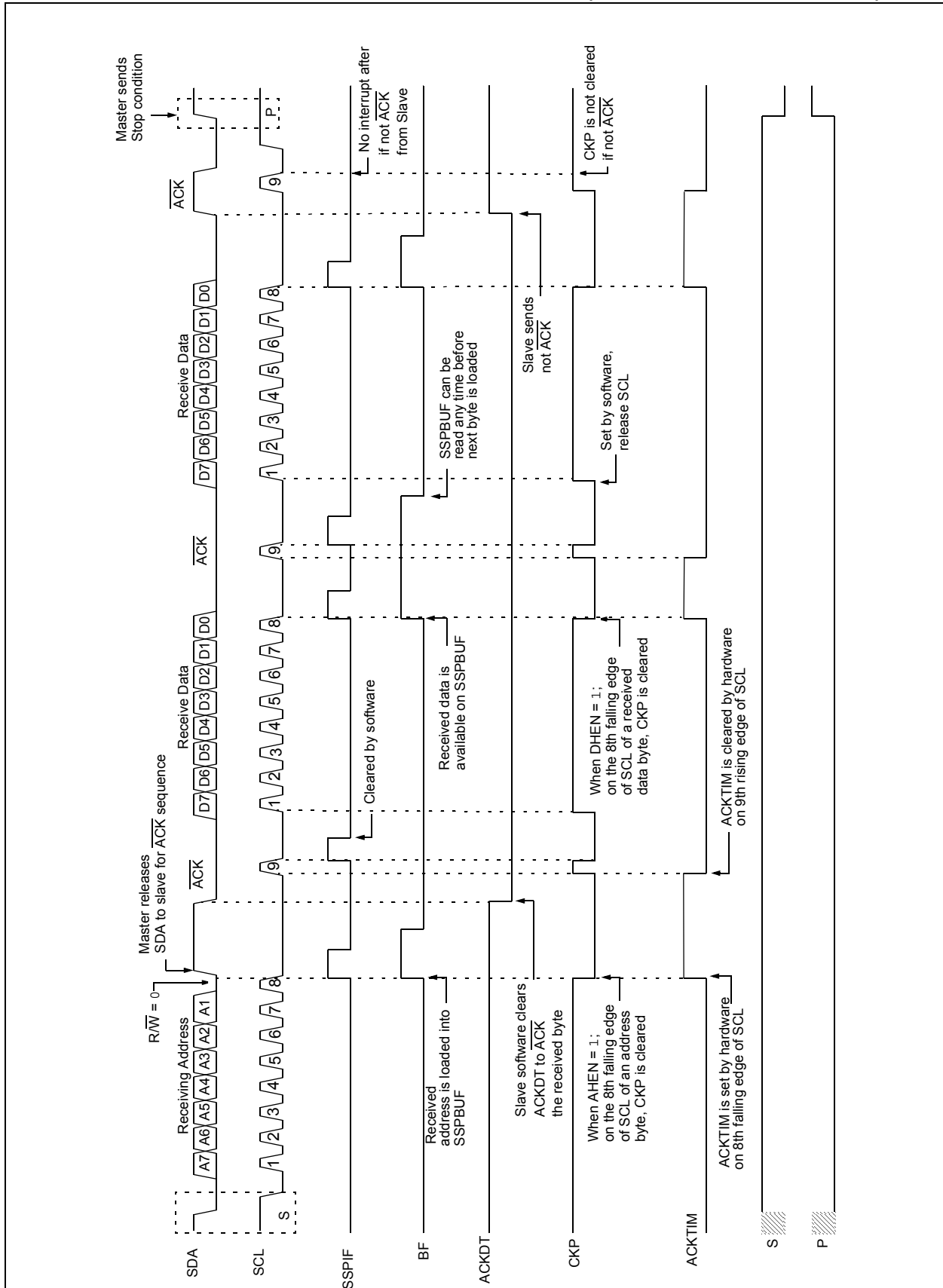

© 2012-2016 Microchip Technology Inc.

FIGURE 20-17: I²C SLAVE, 7-BIT ADDRESS, RECEPTION (SEN = 1, AHEN = 1, DHEN = 1)

20.5.3 SLAVE TRANSMISSION

When the R/$\overline{\text{W}}$ bit of the incoming address byte is set and an address match occurs, the R/$\overline{\text{W}}$ bit of the SSPSTAT register is set. The received address is loaded into the SSPBUF register, and an $\overline{\text{ACK}}$ pulse is sent by the slave on the ninth bit.

Following the $\overline{\text{ACK}}$, slave hardware clears the CKP bit and the SCL pin is held low (see **Section 20.5.6 "Clock Stretching"** for more detail). By stretching the clock, the master will be unable to assert another clock pulse until the slave is done preparing the transmit data.

The transmit data must be loaded into the SSPBUF register which also loads the SSPSR register. Then the SCL pin should be released by setting the CKP bit of the SSPCON1 register. The eight data bits are shifted out on the falling edge of the SCL input. This ensures that the SDA signal is valid during the SCL high time.

The $\overline{\text{ACK}}$ pulse from the master-receiver is latched on the rising edge of the ninth SCL input pulse. This $\overline{\text{ACK}}$ value is copied to the ACKSTAT bit of the SSPCON2 register. If ACKSTAT is set (not $\overline{\text{ACK}}$), then the data transfer is complete. In this case, when the not $\overline{\text{ACK}}$ is latched by the slave, the slave goes Idle and waits for another occurrence of the Start bit. If the SDA line was low ($\overline{\text{ACK}}$), the next transmit data must be loaded into the SSPBUF register. Again, the SCL pin must be released by setting bit CKP.

An MSSP interrupt is generated for each data transfer byte. The SSPIF bit must be cleared by software and the SSPSTAT register is used to determine the status of the byte. The SSPIF bit is set on the falling edge of the ninth clock pulse.

20.5.3.1 Slave Mode Bus Collision

A slave receives a Read request and begins shifting data out on the SDA line. If a bus collision is detected and the SBCDE bit of the SSPCON3 register is set, the BCLIF bit of the PIR register is set. Once a bus collision is detected, the slave goes Idle and waits to be addressed again. User software can use the BCLIF bit to handle a slave bus collision.

20.5.3.2 7-bit Transmission

A master device can transmit a read request to a slave, and then clock data out of the slave. The list below outlines what software for a slave will need to do to accomplish a standard transmission. Figure 20-17 can be used as a reference to this list.

1. Master sends a Start condition on SDA and SCL.
2. S bit of SSPSTAT is set; SSPIF is set if interrupt on Start detect is enabled.
3. Matching address with R/$\overline{\text{W}}$ bit set is received by the Slave setting SSPIF bit.
4. Slave hardware generates an $\overline{\text{ACK}}$ and sets SSPIF.
5. SSPIF bit is cleared by user.
6. Software reads the received address from SSPBUF, clearing BF.
7. R/$\overline{\text{W}}$ is set so CKP was automatically cleared after the $\overline{\text{ACK}}$.
8. The slave software loads the transmit data into SSPBUF.
9. CKP bit is set releasing SCL, allowing the master to clock the data out of the slave.
10. SSPIF is set after the $\overline{\text{ACK}}$ response from the master is loaded into the ACKSTAT register.
11. SSPIF bit is cleared.
12. The slave software checks the ACKSTAT bit to see if the master wants to clock out more data.

> **Note 1:** If the master $\overline{\text{ACK}}$s the clock will be stretched.
>
> **2:** ACKSTAT is the only bit updated on the rising edge of SCL (9th) rather than the falling.

13. Steps 9-13 are repeated for each transmitted byte.
14. If the master sends a not $\overline{\text{ACK}}$; the clock is not held, but SSPIF is still set.
15. The master sends a Restart condition or a Stop.
16. The slave is no longer addressed.

20.5.6 CLOCK STRETCHING

Clock stretching occurs when a device on the bus holds the SCL line low effectively pausing communication. The slave may stretch the clock to allow more time to handle data or prepare a response for the master device. A master device is not concerned with stretching as anytime it is active on the bus and not transferring data it is stretching. Any stretching done by a slave is invisible to the master software and handled by the hardware that generates SCL.

The CKP bit of the SSPCON1 register is used to control stretching in software. Any time the CKP bit is cleared, the module will wait for the SCL line to go low and then hold it. Setting CKP will release SCL and allow more communication.

20.5.6.1 Normal Clock Stretching

Following an $\overline{ACK}$ if the R/$\overline{W}$ bit of SSPSTAT is set, a read request, the slave hardware will clear CKP. This allows the slave time to update SSPBUF with data to transfer to the master. If the SEN bit of SSPCON2 is set, the slave hardware will always stretch the clock after the $\overline{ACK}$ sequence. Once the slave is ready; CKP is set by software and communication resumes.

> **Note 1:** The BF bit has no effect on if the clock will be stretched or not. This is different than previous versions of the module that would not stretch the clock, clear CKP, if SSPBUF was read before the 9th falling edge of SCL.
>
> **2:** Previous versions of the module did not stretch the clock for a transmission if SSPBUF was loaded before the 9th falling edge of SCL. It is now always cleared for read requests.

20.5.6.2 10-bit Addressing Mode

In 10-bit Addressing mode, when the UA bit is set the clock is always stretched. This is the only time, the SCL is stretched without CKP being cleared. SCL is released immediately after a write to SSPADD.

> **Note:** Previous versions of the module did not stretch the clock if the second address byte did not match.

20.5.6.3 Byte NACKing

When AHEN bit of SSPCON3 is set; CKP is cleared by hardware after the 8th falling edge of SCL for a received matching address byte. When DHEN bit of SSPCON3 is set; CKP is cleared after the 8th falling edge of SCL for received data.

Stretching after the 8th falling edge of SCL allows the slave to look at the received address or data and decide if it wants to ACK the received data.

20.5.7 CLOCK SYNCHRONIZATION AND THE CKP BIT

Any time the CKP bit is cleared, the module will wait for the SCL line to go low and then hold it. However, clearing the CKP bit will not assert the SCL output low until the SCL output is already sampled low. Therefore, the CKP bit will not assert the SCL line until an external I^{2}C master device has already asserted the SCL line. The SCL output will remain low until the CKP bit is set and all other devices on the I^{2}C bus have released SCL. This ensures that a write to the CKP bit will not violate the minimum high time requirement for SCL (see Figure 20-22).

FIGURE 20-23: CLOCK SYNCHRONIZATION TIMING
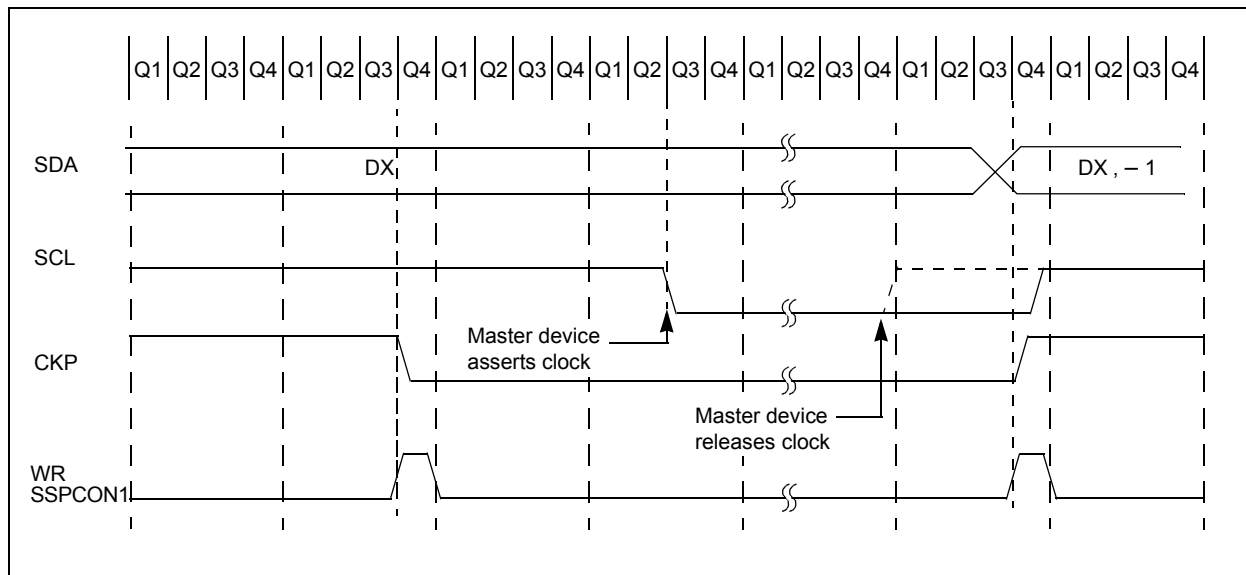
21.1 Capture Mode

The Capture mode function described in this section is available and identical for all CCP modules.

Capture mode makes use of the 16-bit Timer1 resource. When an event occurs on the CCPx pin, the 16-bit CCPRxH:CCPRxL register pair captures and stores the 16-bit value of the TMR1H:TMR1L register pair, respectively. An event is defined as one of the following and is configured by the CCPxM<3:0> bits of the CCPxCON register:

• Every falling edge
• Every rising edge
• Every 4th rising edge
• Every 16th rising edge

When a capture is made, the Interrupt Request Flag bit CCPxIF of the PIRx register is set. The interrupt flag must be cleared in software. If another capture occurs before the value in the CCPRxH, CCPRxL register pair is read, the old captured value is overwritten by the new captured value.

Figure 21-1 shows a simplified diagram of the Capture operation.

21.1.1 CCP PIN CONFIGURATION

In Capture mode, the CCPx pin should be configured as an input by setting the associated TRIS control bit.

Also, the CCP2 pin function can be moved to alternative pins using the APFCON register. Refer to **Section Register 12-1: "APFCON: Alternate Pin Function Control Register"** for more details.

Note: If the CCPx pin is configured as an output, a write to the port can cause a capture condition.

FIGURE 21-1: CAPTURE MODE OPERATION BLOCK DIAGRAM
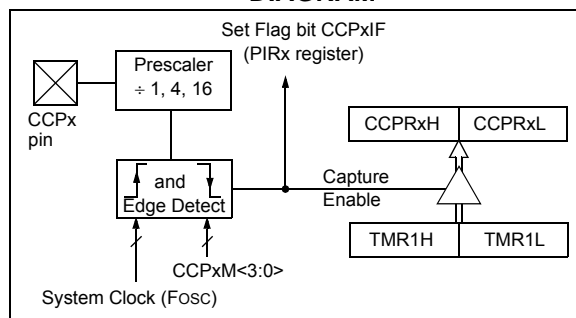
21.1.2 TIMER1 MODE RESOURCE

Timer1 must be running in Timer mode or Synchronized Counter mode for the CCP module to use the capture feature. In Asynchronous Counter mode, the capture operation may not work.

See **Section 18.0 "Timer1 Module with Gate Control"** for more information on configuring Timer1.

21.1.3 SOFTWARE INTERRUPT MODE

When the Capture mode is changed, a false capture interrupt may be generated. The user should keep the CCPxIE interrupt enable bit of the PIEx register clear to avoid false interrupts. Additionally, the user should clear the CCPxIF interrupt flag bit of the PIRx register following any change in Operating mode.

21.1.4 CCP PRESCALER

There are four prescaler settings specified by the CCPxM<3:0> bits of the CCPxCON register. Whenever the CCP module is turned off, or the CCP module is not in Capture mode, the prescaler counter is cleared. Any Reset will clear the prescaler counter.

Switching from one capture prescaler to another does not clear the prescaler and may generate a false interrupt. To avoid this unexpected operation, turn the module off by clearing the CCPxCON register before changing the prescaler. Equation 21-1 demonstrates the code to perform this function.

EXAMPLE 21-1: CHANGING BETWEEN CAPTURE PRESCALERS

```
BANKSEL CCPxCON     ;Set Bank bits to point
                    ;to CCPxCON
CLRF    CCPxCON     ;Turn CCP module off
MOVLW   NEW_CAPT_PS ;Load the W reg with
                    ;the new prescaler
                    ;move value and CCP ON
MOVWF   CCPxCON     ;Load CCPxCON with this
                    ;value
```

PIC16(L)F1512/3

21.2.5 COMPARE DURING SLEEP

The Compare mode is dependent upon the system clock (FOSC) for proper operation. Since FOSC is shut down during Sleep mode, the Compare mode will not function properly during Sleep.

21.2.6 ALTERNATE PIN LOCATIONS

This module incorporates I/O pins that can be moved to other locations with the use of the alternate pin function register APFCON. To determine which pins can be moved and what their default locations are upon a Reset, see **Section 12.1 "Alternate Pin Function"** for more information.

21.3 PWM Overview

Pulse-Width Modulation (PWM) is a scheme that provides power to a load by switching quickly between fully on and fully off states. The PWM signal resembles a square wave where the high portion of the signal is considered the on state and the low portion of the signal is considered the off state. The high portion, also known as the pulse width, can vary in time and is defined in steps. A larger number of steps applied, which lengthens the pulse width, also supplies more power to the load. Lowering the number of steps applied, which shortens the pulse width, supplies less power. The PWM period is defined as the duration of one complete cycle or the total amount of on and off time combined.

PWM resolution defines the maximum number of steps that can be present in a single PWM period. A higher resolution allows for more precise control of the pulse width time and in turn the power that is applied to the load.

The term duty cycle describes the proportion of the on time to the off time and is expressed in percentages, where 0% is fully off and 100% is fully on. A lower duty cycle corresponds to less power applied and a higher duty cycle corresponds to more power applied.

Figure 21-3 shows a typical waveform of the PWM signal.

21.3.1 STANDARD PWM OPERATION

The standard PWM function described in this section is available and identical for all CCP modules.

The standard PWM mode generates a Pulse-Width Modulation (PWM) signal on the CCPx pin with up to 10 bits of resolution. The period, duty cycle, and resolution are controlled by the following registers:

- PR2 registers
- T2CON registers
- CCPRxL registers
- CCPxCON registers

Figure 21-4 shows a simplified block diagram of PWM operation.

Note 1: The corresponding TRIS bit must be cleared to enable the PWM output on the CCPx pin.

2: Clearing the CCPxCON register will relinquish control of the CCPx pin.
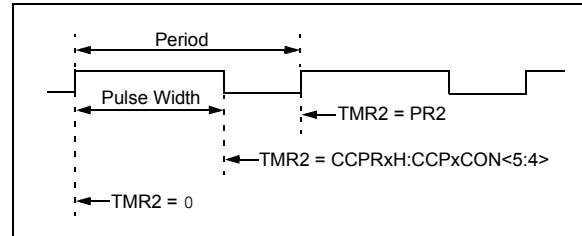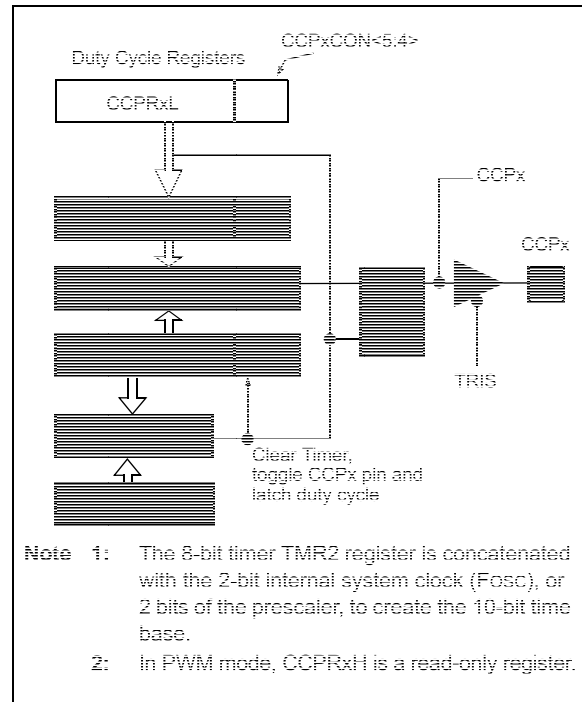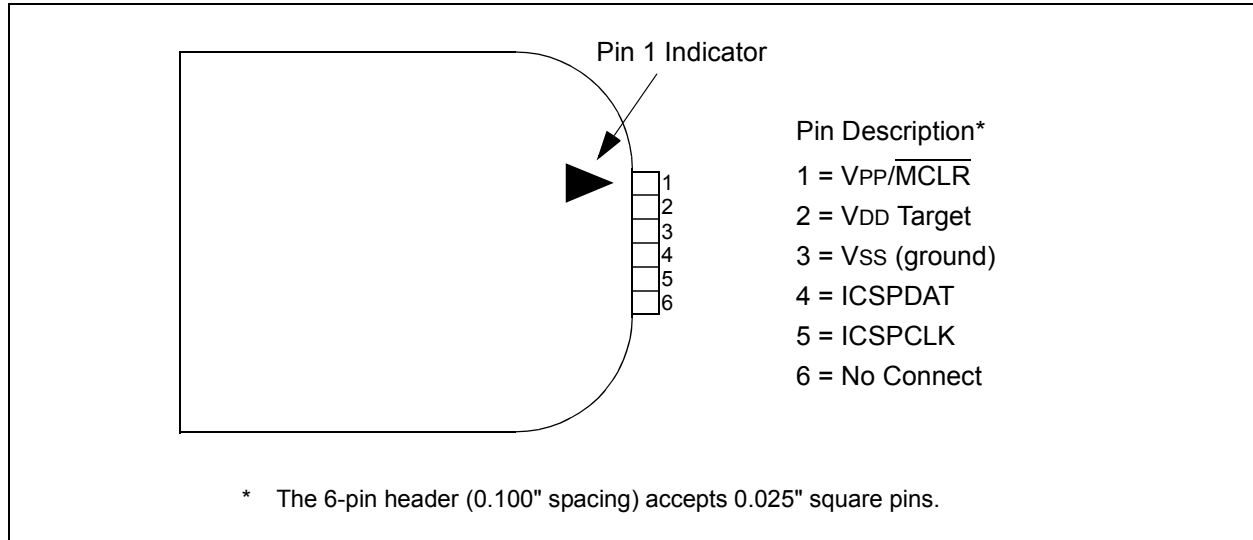

FIGURE 21-3:   CCP PWM OUTPUT SIGNAL

FIGURE 21-4:   SIMPLIFIED PWM BLOCK DIAGRAM

Note 1: The 8-bit timer TMR2 register is concatenated with the 2-bit internal system clock (FOSC), or 2 bits of the prescaler, to create the 10-bit time base.

2: In PWM mode, CCPRxH is a read-only register.

Another connector often found in use with the PICkit™ programmers is a standard 6-pin header with 0.1 inch spacing. Refer to Figure 23-2.
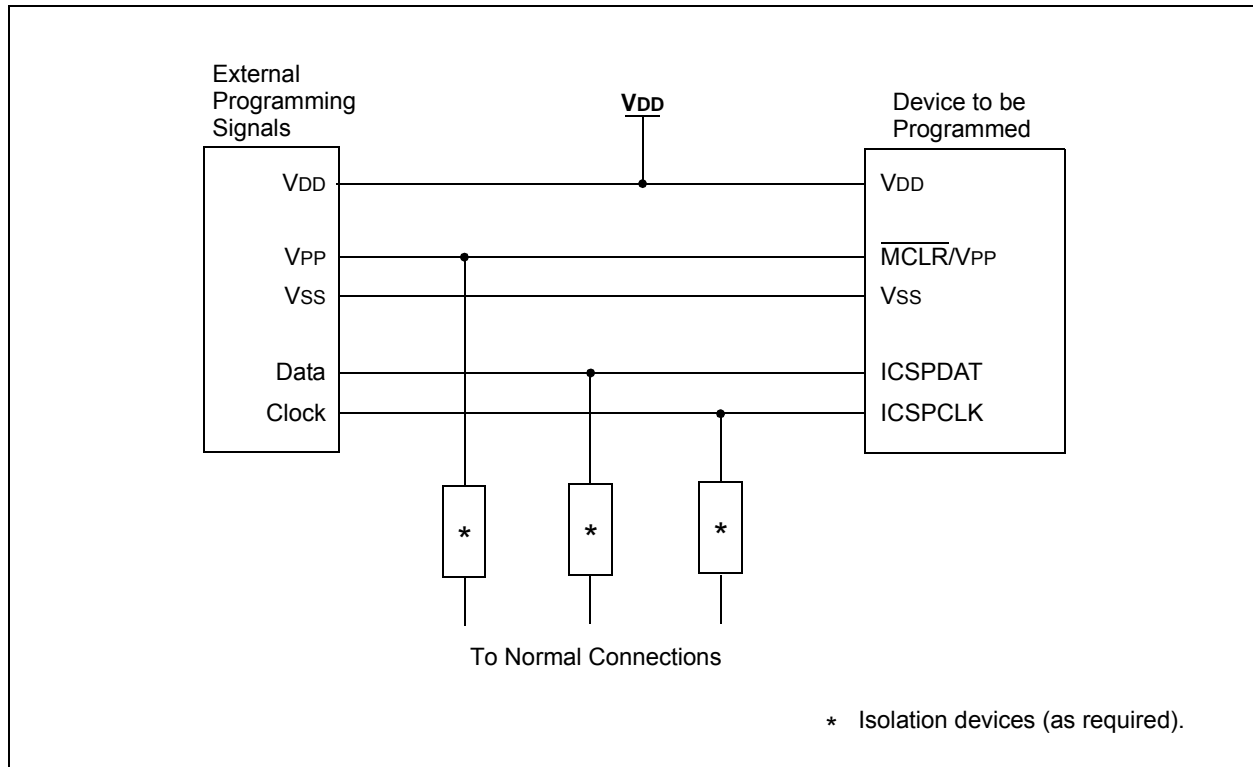
FIGURE 23-2:   PICkit™ PROGRAMMER STYLE CONNECTOR INTERFACE

Pin 1 Indicator

Pin Description*

1 = VPP/$\overline{\text{MCLR}}$

2 = VDD Target

3 = VSS (ground)

4 = ICSPDAT

5 = ICSPCLK

6 = No Connect

*   The 6-pin header (0.100" spacing) accepts 0.025" square pins.

For additional interface recommendations, refer to your specific device programmer manual prior to PCB design.

It is recommended that isolation devices be used to separate the programming pins from other circuitry. The type of isolation is highly dependent on the specific application and may include devices such as resistors, diodes, or even jumpers. See Figure 23-3 for more information.

FIGURE 23-3:   TYPICAL CONNECTION FOR ICSP™ PROGRAMMING

External Programming Signals

VDD

Device to be Programmed

VDD — VDD

VPP — $\overline{\text{MCLR}}$/VPP

VSS — VSS

Data — ICSPDAT

Clock — ICSPCLK

To Normal Connections

*   Isolation devices (as required).

24.2 Instruction Descriptions

ADDFSR	Add Literal to FSRn
Syntax:	[*label*] ADDFSR FSRn, k
Operands:	-32 ≤ k ≤ 31 n ∈ [0, 1]
Operation:	FSR(n) + k → FSR(n)
Status Affected:	None
Description:	The signed 6-bit literal 'k' is added to the contents of the FSRnH:FSRnL register pair. FSRn is limited to the range 0000h - FFFFh. Moving beyond these bounds will cause the FSR to wrap-around.

ADDLW	Add literal and W
Syntax:	[*label*] ADDLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) + k → (W)
Status Affected:	C, DC, Z
Description:	The contents of the W register are added to the 8-bit literal 'k' and the result is placed in the W register.

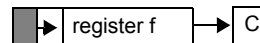| ADDWF	Add W and f
Syntax:	[*label*] ADDWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) + (f) → (destination)
Status Affected:	C, DC, Z
Description:	Add the contents of the W register with register 'f'. If 'd' is '0', the result is stored in the W register. If 'd' is '1', the result is stored back in register 'f'.

ADDWFC	ADD W and CARRY bit to f
Syntax:	[*label*] ADDWFC f {,d}
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) + (f) + (C) → dest
Status Affected:	C, DC, Z
Description:	Add W, the Carry flag and data memory location 'f'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed in data memory location 'f'.

ANDLW	AND literal with W
Syntax:	[*label*] ANDLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) .AND. (k) → (W)
Status Affected:	Z
Description:	The contents of W register are AND'ed with the 8-bit literal 'k'. The result is placed in the W register.

ANDWF	AND W with f
Syntax:	[*label*] ANDWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) .AND. (f) → (destination)
Status Affected:	Z
Description:	AND the W register with register 'f'. If 'd' is '0', the result is stored in the W register. If 'd' is '1', the result is stored back in register 'f'.

ASRF	Arithmetic Right Shift
Syntax:	[*label*] ASRF f {,d}
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f<7>)→ dest<7> (f<7:1>) → dest<6:0>, (f<0>) → C,
Status Affected:	C, Z
Description:	The contents of register 'f' are shifted one bit to the right through the Carry flag. The MSb remains unchanged. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is stored back in register 'f'.

CALL	Call Subroutine
Syntax:	[*label*] CALL k
Operands:	$0 \leq k \leq 2047$
Operation:	(PC)+ 1$\rightarrow$ TOS, k $\rightarrow$ PC<10:0>, (PCLATH<6:3>) $\rightarrow$ PC<14:11>
Status Affected:	None
Description:	Call Subroutine. First, return address (PC + 1) is pushed onto the stack. The 11-bit immediate address is loaded into PC bits <10:0>. The upper bits of the PC are loaded from PCLATH. CALL is a 2-cycle instruction.

CALLW	Subroutine Call With W
Syntax:	[*label*] CALLW
Operands:	None
Operation:	(PC) +1 $\rightarrow$ TOS, (W) $\rightarrow$ PC<7:0>, (PCLATH<6:0>) $\rightarrow$ PC<14:8>
Status Affected:	None
Description:	Subroutine call with W. First, the return address (PC + 1) is pushed onto the return stack. Then, the contents of W is loaded into PC<7:0>, and the contents of PCLATH into PC<14:8>. CALLW is a 2-cycle instruction.

CLRF	Clear f
Syntax:	[*label*] CLRF f
Operands:	$0 \leq f \leq 127$
Operation:	00h $\rightarrow$ (f) 1 $\rightarrow$ Z
Status Affected:	Z
Description:	The contents of register 'f' are cleared and the Z bit is set.

CLRW	Clear W
Syntax:	[*label*] CLRW
Operands:	None
Operation:	00h $\rightarrow$ (W) 1 $\rightarrow$ Z
Status Affected:	Z
Description:	W register is cleared. Zero bit (Z) is set.

CLRWDT	Clear Watchdog Timer
Syntax:	[*label*] CLRWDT
Operands:	None
Operation:	00h $\rightarrow$ WDT 0 $\rightarrow$ WDT prescaler, 1 $\rightarrow$ $\overline{\text{TO}}$ 1 $\rightarrow$ $\overline{\text{PD}}$
Status Affected:	$\overline{\text{TO}}$, $\overline{\text{PD}}$
Description:	CLRWDT instruction resets the Watchdog Timer. It also resets the prescaler of the WDT. Status bits $\overline{\text{TO}}$ and $\overline{\text{PD}}$ are set.

COMF	Complement f
Syntax:	[*label*] COMF f,d
Operands:	$0 \leq f \leq 127$ d $\in$ [0,1]
Operation:	($\overline{\text{f}}$) $\rightarrow$ (destination)
Status Affected:	Z
Description:	The contents of register 'f' are complemented. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f'.

DECF	Decrement f
Syntax:	[*label*] DECF f,d
Operands:	$0 \leq f \leq 127$ d $\in$ [0,1]
Operation:	(f) - 1 $\rightarrow$ (destination)
Status Affected:	Z
Description:	Decrement register 'f'. If 'd' is '0', the result is stored in the W register. If 'd' is '1', the result is stored back in register 'f'.

RRF	Rotate Right f through Carry
Syntax:	[*label*] RRF f,d
Operands:	$0 \leq f \leq 127$ $d \in [0,1]$
Operation:	See description below
Status Affected:	C
Description:	The contents of register 'f' are rotated one bit to the right through the Carry flag. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is placed back in register 'f'.
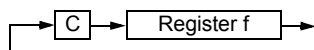
SLEEP	Enter Sleep mode
Syntax:	[*label*] SLEEP
Operands:	None
Operation:	$00h \rightarrow WDT$, $0 \rightarrow \overline{WDT}$ prescaler, $1 \rightarrow \overline{TO}$, $0 \rightarrow \overline{PD}$
Status Affected:	$\overline{TO}$, $\overline{PD}$
Description:	The power-down Status bit, $\overline{PD}$ is cleared. Time-out Status bit, $\overline{TO}$ is set. Watchdog Timer and its prescaler are cleared. The processor is put into Sleep mode with the oscillator stopped.

SUBLW	Subtract W from literal
Syntax:	[*label*] SUBLW k
Operands:	$0 \leq k \leq 255$
Operation:	$k - (W) \rightarrow (W)$
Status Affected:	C, DC, Z
Description:	The W register is subtracted (2's complement method) from the 8-bit literal 'k'. The result is placed in the W register.

C = 0	W > k
C = 1	W ≤ k
DC = 0	W<3:0> > k<3:0>
DC = 1	W<3:0> ≤ k<3:0>

SUBWF	Subtract W from f
Syntax:	[*label*] SUBWF f,d
Operands:	$0 \leq f \leq 127$ $d \in [0,1]$
Operation:	$(f) - (W) \rightarrow$ (destination)
Status Affected:	C, DC, Z
Description:	Subtract (2's complement method) W register from register 'f'. If 'd' is '0', the result is stored in the W register. If 'd' is '1', the result is stored back in register 'f'.

C = 0	W > f
C = 1	W ≤ f
DC = 0	W<3:0> > f<3:0>
DC = 1	W<3:0> ≤ f<3:0>

| SUBWFB	Subtract W from f with Borrow
Syntax:	SUBWFB f {,d}
Operands:	$0 \leq f \leq 127$ $d \in [0,1]$
Operation:	$(f) - (W) - (\overline{B}) \rightarrow$ dest
Status Affected:	C, DC, Z
Description:	Subtract W and the BORROW flag (CARRY) from register 'f' (2's complement method). If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f'.

28-Lead Plastic Shrink Small Outline (SS) – 5.30 mm Body [SSOP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging
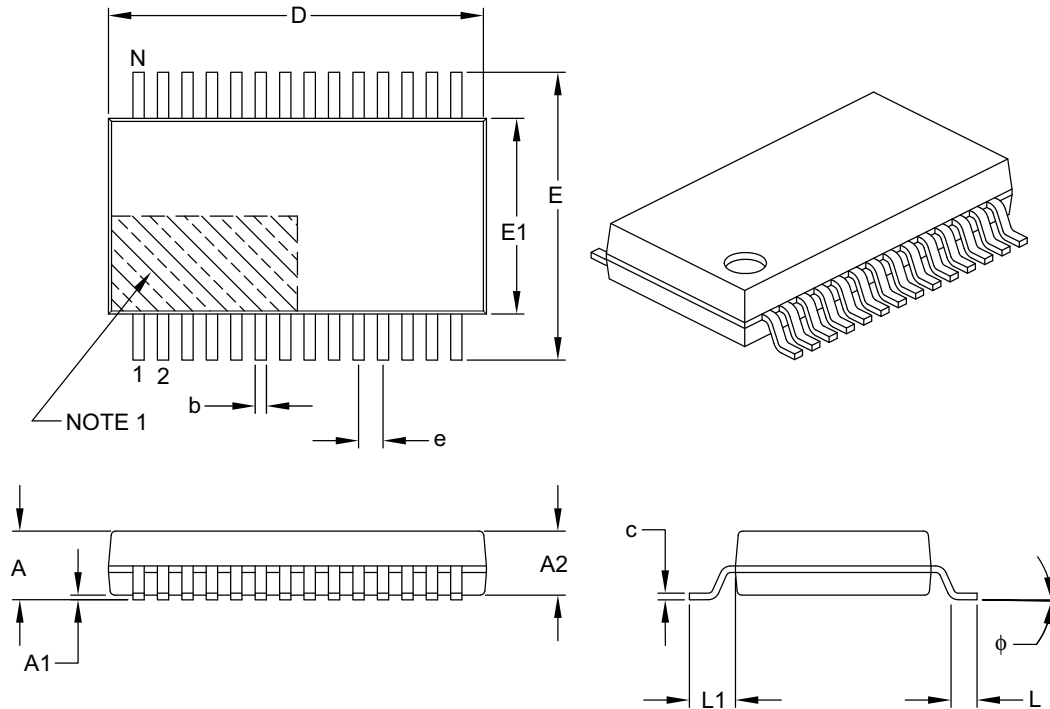
Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N		28	
Pitch	e		0.65 BSC	
Overall Height	A	–	–	2.00
Molded Package Thickness	A2	1.65	1.75	1.85
Standoff	A1	0.05	–	–
Overall Width	E	7.40	7.80	8.20
Molded Package Width	E1	5.00	5.30	5.60
Overall Length	D	9.90	10.20	10.50
Foot Length	L	0.55	0.75	0.95
Footprint	L1		1.25 REF	
Lead Thickness	c	0.09	–	0.25
Foot Angle	φ	0°	4°	8°
Lead Width	b	0.22	–	0.38

Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.20 mm per side.
3. Dimensioning and tolerancing per ASME Y14.5M.

   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-073B